UNIVERSIDAD DE OVIEDO
Departamento de Ingeniería Eléctrica,
Electrónica, de Computadores y Sistemas

INGENIERÍA DE SISTEMAS Y AUTOMÁTICA

BeagleBone con Arch Linux

Manual de instalación y manejo

Ignacio Alvarez García – Noviembre 2012

INDICE

1. Instalación del Sistema Operativo Arch Linux 1
1.1. Creación de tarjeta SD 1
1.1.1. Crear dos particiones en la tarjeta SD: 1
1.1.2. Descargar e instalar archivos del S.O. 1
1.2. Configuración del arranque 2
1.2.1. Establecer dirección IP estática 2
1.2.2. Configurar uso de ciertos pines 2
1.3. Añadir programas de interés 2
1.3.1. Instalar GDB remoto 2
1.4. Instalar, probar y terminar configuración 3
2. Configurar proyecto eclipse 5
2.1. Pasos previos 5
2.1.1. Toolchain 5
2.1.2. Sistema remoto en eclipse 5
2.2. Creación de proyecto para ejecución en Beaglebone 7
2.2.1. Paso 1: creación de proyecto en modo local 7
2.2.2. Paso 2: configuración de compilación para BeagleBone 8
2.2.3. Paso 3: configuración de ejecución remota en BeagleBone 9
2.2.4. Paso 4: instalación definitiva y arranque automático 10
3. Acceso a dispositivos de E/S 11
3.1. Interfaz hardware 11
3.1.1. Conectores de expansión P8 y P9 11
3.1.2. E/S digitales 14
3.1.3. Entradas analógicas 14
3.1.4. Salidas PWM 14
3.2. Acceso a la E/S mediante Linux fs (filesystem) 15
3.2.1. Manejo de E/S digital en pines P8 y P9 y LEDs de placa 16
3.2.2. Manejo de entradas analógicas 16

3.2.3. Manejo de salidas PWM 17
3.3. Librería de E/S 17
3.3.1. Configuración de proyecto 17
3.3.1. Manejo de E/S digital 19
3.3.2. Manejo de entrada analógica 20
3.3.3. Manejo de salida PWM 21
3.4. Fuentes de documentación 22

1. Instalación del Sistema Operativo Arch Linux

(de: http://archlinuxarm.org/platforms/armv7/beaglebone)

Estos pasos ya están realizados en la tarjeta SD disponible.

1.1. Creación de tarjeta SD

Ejecutar un S.O. Linux en el PC y comprobar en qué dispositivo se encuentra la tarjeta SD (habitualmente /dev/mmcblk0, valor usado en los ejemplos).

Arrancar una ventana de terminal, y crear un directorio de trabajo:

```
$ mkdir ArchLinuxInstall
$ cd ArchLinuxInstall
```

1.1.1. Crear dos particiones en la tarjeta SD:

- La primera partición debe ser FAT16 LBA (tipo de partición 0E), con un tamaño de 128Mb. Marcar esta partición como bootable.
- La segunda partición debe ser tipo Linux (ext3), con el resto del espacio de la SD (mínimo 1.2Gb).
- Formatear la primera partición con mkfs.vfat:

```
$ sudo mkfs.vfat –F 16 –n "bootloader" /dev/mmcblk0p1
```

- Formatear la segunda partición con mkfs.ext3:

```
$ sudo mkfs.ext3 -L "rootfs" /dev/mmcblk0p2
```

1.1.2. Descargar e instalar archivos del S.O.

- Descargar en el directorio temporal el archivo comprimido Beaglebone bootloader (http://archlinuxarm.org/os/omap/BeagleBone-bootloader.tar.gz) y extraer los archivos en la primera partición de la SD:

```
$ mkdir temp_part1
$ sudo mount /dev/mmcblk0p1 temp_part1
$ cd temp_part1
$ sudo tar –xvf ../BeagleBone-bootloader.tar.gz
$ cd ..
```

- Descargar en el directorio temporal el archivo comprimido root filesystem (http://archlinuxarm.org/os/ArchLinuxARM-am33x-latest.tar.gz) y extraer los archivos en la segunda partición:

```
$ mkdir temp_part2
$ sudo mount /dev/mmcblk0p2 temp_part2
$ cd temp_part2
$ sudo tar –xvf ../ArchLinuxARM-am33x-latest.tar.gz
$ cd ..
```

- Copiar el archivo /boot/uImage de la segunda a la primera partición:

```
$ sudo cp temp_part2/boot/uImage temp_part1/boot
```

1.2. Configuración del arranque

1.2.1. Establecer dirección IP estática

- Editar temp_part2/etc/rc.conf para incluir una dirección IP estática (ejemplo: 192.168.100.1, máscara de subred 255.255.255.0):

```
# ------------------------------------------------------------------------
# NETWORKING
# ------------------------------------------------------------------------
...
interface=eth0
address=192.168.100.1
netmask=255.255.255.0
broadcast=192.168.100.255
gateway=
```

1.2.2. Configurar uso de ciertos pines

- Editar el archivo temp_part2/etc/rc.local de la partición 2:

```
#!/bin/bash
#
# /etc/rc.local: Local multi-user start-up script.
#
mount -t debugfs none /sys/kernel/debug/
modprobe ti_tscadc
# AQUI: añadir otros comandos de arranque automático (ej. programa de usuario)
```

1.3. Añadir programas de interés

1.3.1. Instalar GDB remoto

(De http://brain-dump.org/blog/entry/138)

- Instalar parte a ejecutar en PC (gdb-multiarch):

```
$ sudo apt-get install gdb-multiarch
```

- Instalar parte a ejecutar en BeagleBone (gdbserver):
  - Obtener fuentes (http://ftp.gul.es/gnu/ftp.gnu.org/gnu/gdb/).
  - Descomprimir y compilar:

```
$ tar –xvf gdb-7.5.tar.gz
$ cd gdb-7.5/gdb/gdbserver
$ ./configure --target=arm-bcm2708hardfp-linux-gnueabi --host=arm-bcm2708hardfp-linux-gnueabi --prefix=/usr --program-prefix=""
$ make
$ make DESTDIR=/home/developer/RaspberryPi/gdbserverforarm install
```

```
$ cd ..
```

- Copiar a BeableBone:

```
$ sudo cp /home/developer/RaspberryPi/gdbserverforarm/usr/bin/gdbserver
  temp_part2/usr/bin
```

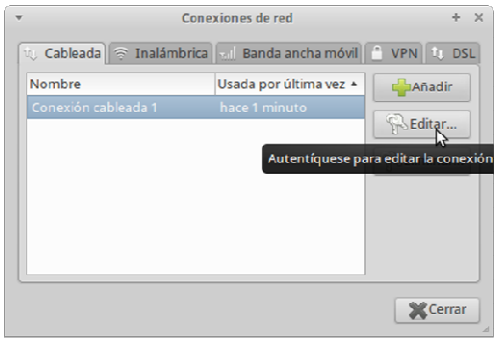
- Entrar en consola BeagleBone y cambiar permisos a /usr/bin/gdbserver, según lo indicado en el apartado siguiente.

1.4. Instalar, probar y terminar configuración

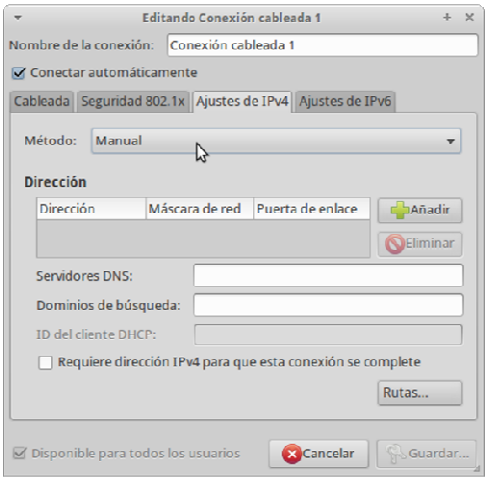
- Desmontar las dos particiones:

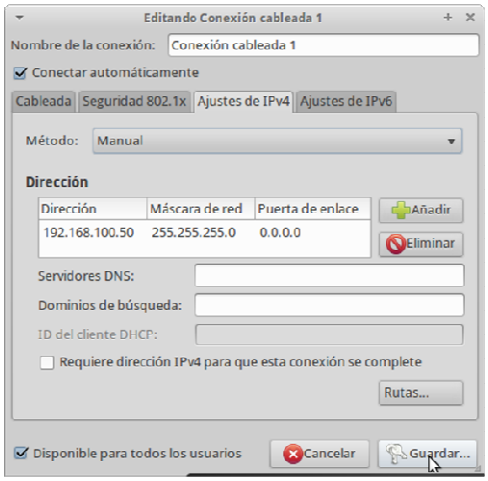
```
$ sudo umount temp_part1
$ sudo umount temp_part2
```

- Desmontar de forma segura la tarjeta SD.
- Configurar conexión de red con dirección estática en S.O. Linux:
  - Asegurar que se utiliza conexión de red con adaptador puente en la configuración de VirtualBox (apartado 1.4 en documento de instalación de Linux bajo VirtualBox).
  - Editar la conexión de red cableada pulsando el icono de red en la parte superior derecha de la consola, seleccionando la conexión de red y pulsando Editar.



  - En las propiedades de la conexión de red, patilla "Ajustes de IPV4", seleccionar Manual y pulsar "Añadir dirección".
  - Añadir una dirección en el rango 192.168.100.xx (excepto la 1), con máscara de red 255.255.255.0. Pulsar guardar y cerrar.





- Instalar tarjeta SD en BeagleBone, conectar cable de red y alimentar.
- Esperar unos segundos y comprobar conexión:

  ```
  $ ping 192.168.100.1
  ```

- Comprobar consola BeagleBone:

  ```
  $ ssh root@192.168.100.1
  Pass: root
  #
  ```

  - Dentro de consola BeagleBone, cambiar permisos a gdbserver:

    ```
    # chmod a+x /usr/bin/gdbserver
    ```

2. Configurar proyecto eclipse

2.1. Pasos previos

Estos pasos ya están realizados en el S.O. XUbuntu disponible.

2.1.1. Toolchain

(de http://www.raspberrypi.org/phpBB3/viewtopic.php?f=31&t=10739)

- Descargar e instalar:

  $ sudo mkdir /opt/RaspberryPi

  $ cd /opt/RaspberryPi

  $ sudo git clone https://github.com/raspberrypi/tools.git

  (ojo con proxies en /etc/environment)

- Probar compilación:

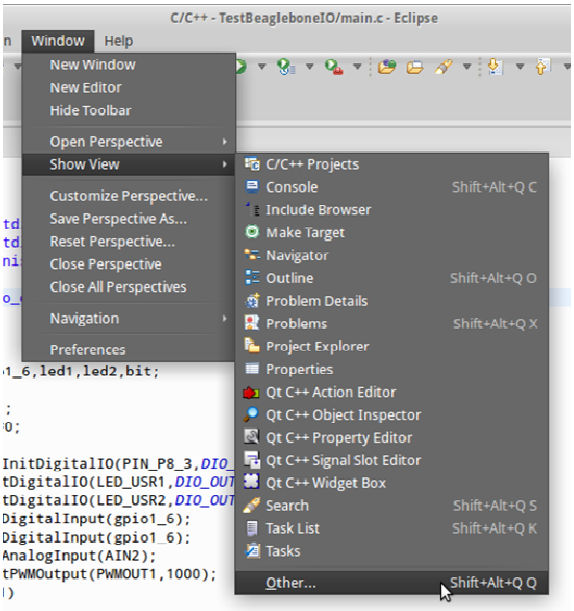
  $ /opt/RaspberryPi/tools/arm-bcm2708/arm-bcm2708hardfp-linux-gnueabi/bin/arm-bcm2708hardfp-linux-gnueabi-g++ -o helloworld.bin helloworld.cpp

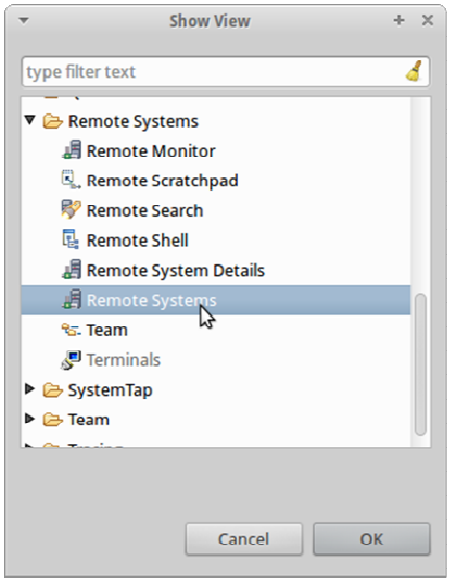
  $ file helloworld.bin -> debe responder elf32 para ARM

2.1.2. Sistema remoto en eclipse

- Entrar en eclipse y seleccionar:

  Window -> Show View -> Other -> Remote Systems -> Remote Systems



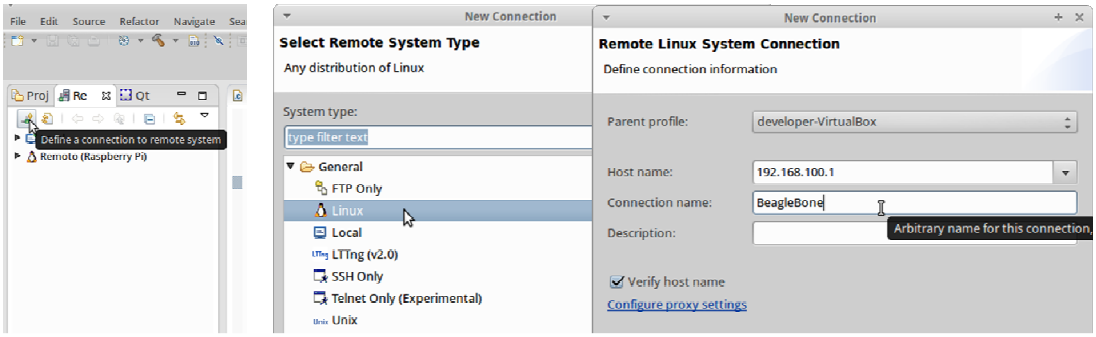


- En pestaña Remote Systems: Local -> new connection:

  Linux

  Host name: dirección IP Raspberry (192.168.100.1)

  Nombre conexión: BeagleBone

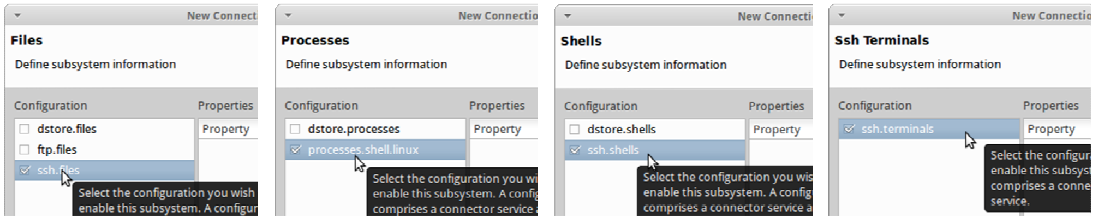


Config -> Files -> ssh.files

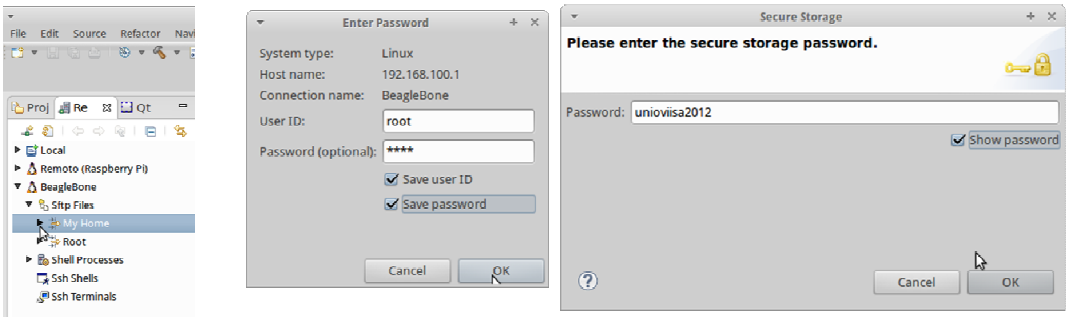
Processes -> processes.shell.linux

Shells -> ssh.shells

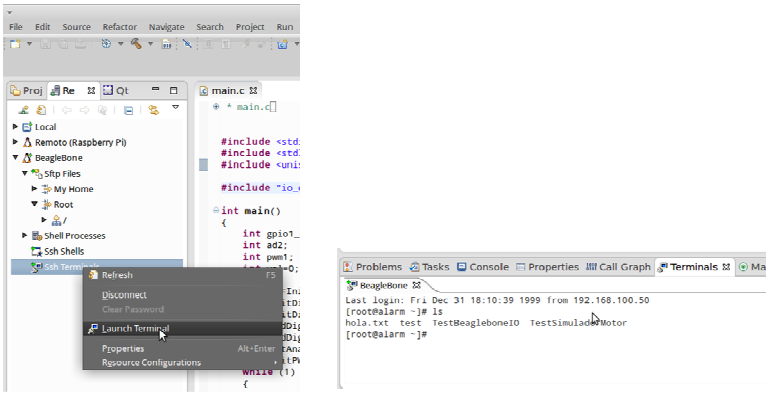
Ssh terminals -> ssh.terminals



- En patilla Sftp Files. Abrir my home. Pide user id y password: root / root. Chequear opciones Save -> Master password (unioviisa2012)



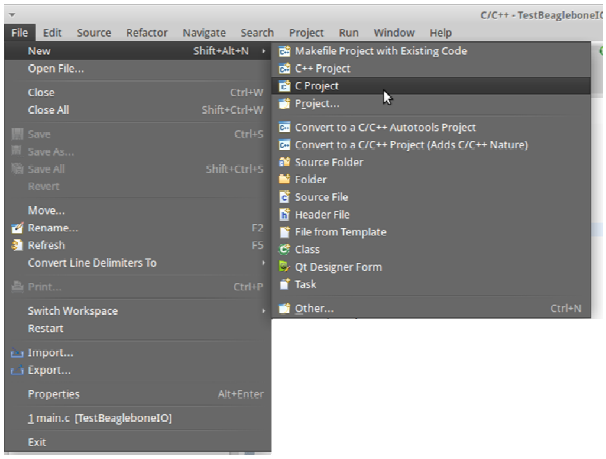
- En patilla SSh Terminals -> Launch terminal. En el prompt # se está trabajando sobre el BeagleBone (ej: ls = listar archivos en el directorio).

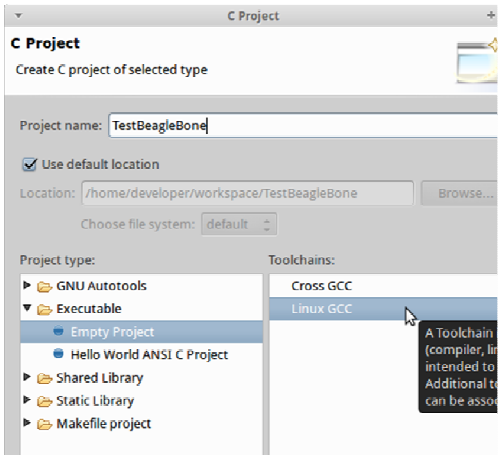


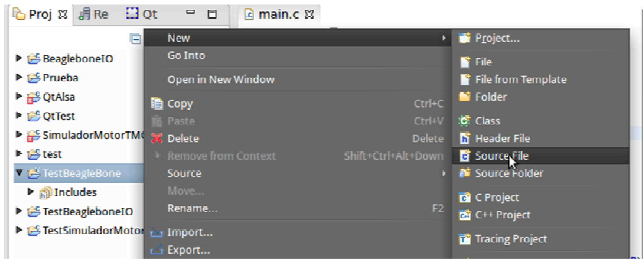
2.2. Creación de proyecto para ejecución en Beaglebone

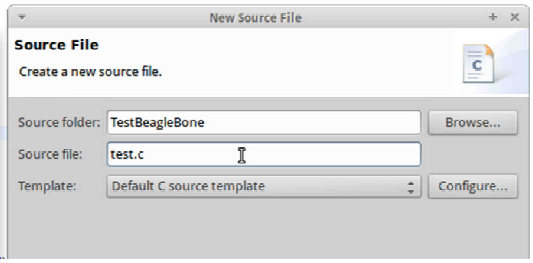
2.2.1. Paso 1: creación de proyecto en modo local

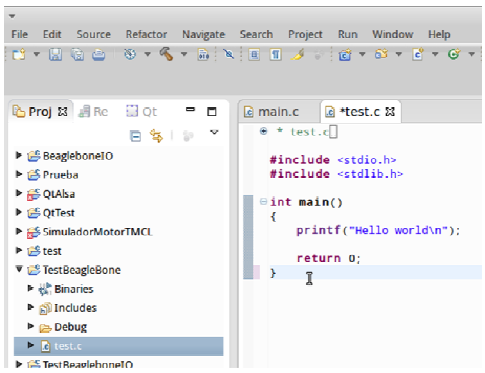
- Crear un proyecto sencillo en C con compilación Linux GCC, y un archivo de código fuente sencillo. Probar compilación y ejecución en equipo local (PC).

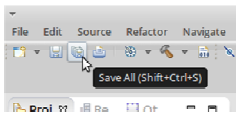


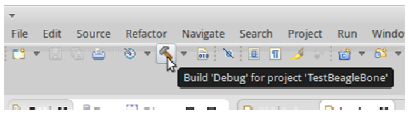


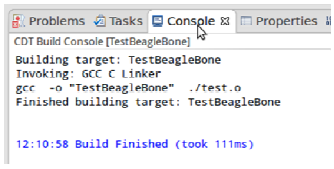


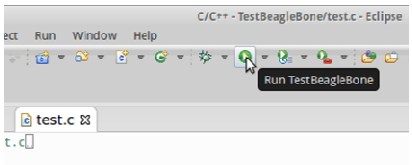


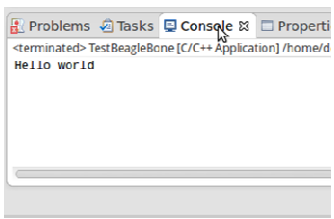






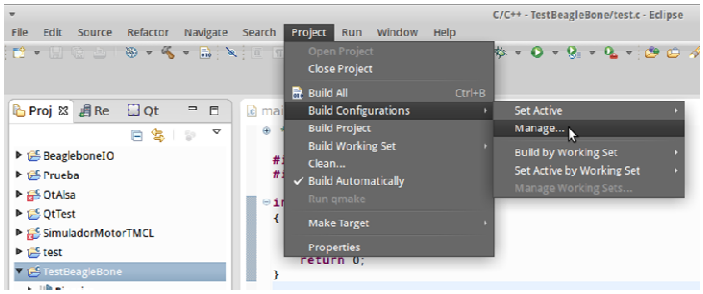

Problems Tasks Console Properties
CDT Build Console [TestBeagleBone]
Building target: TestBeagleBone
Invoking: GCC C Linker
gcc -o "TestBeagleBone" ./test.o
Finished building target: TestBeagleBone
12:10:58 Build Finished (took 111ms)

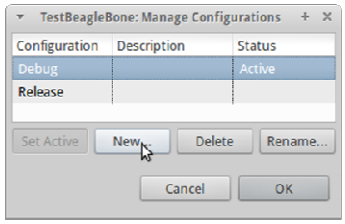


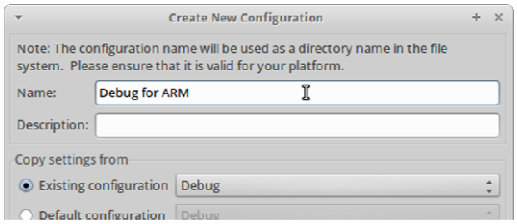


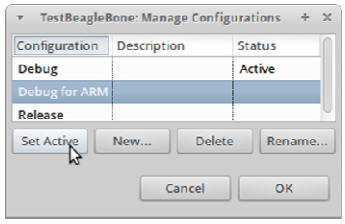
2.2.2. Paso 2: configuración de compilación para BeagleBone

- En Project -> Build configurations -> Manage: añadir y activar configuración Debug para ARM (procesador del BeagleBone).

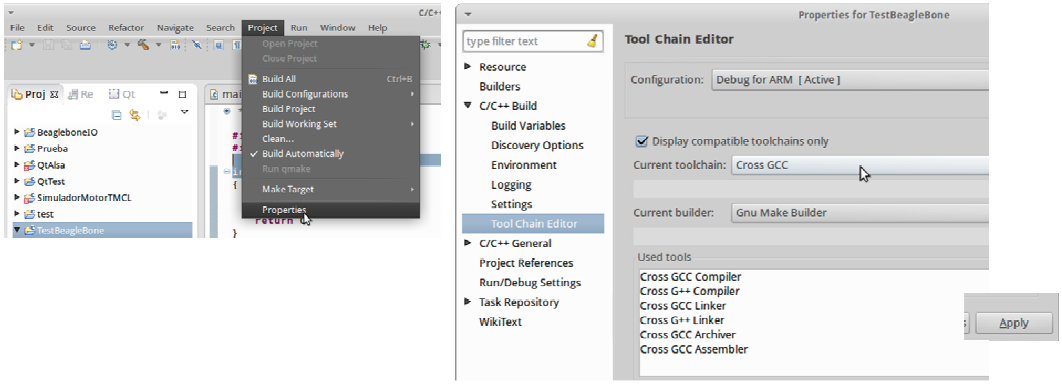





Create New Configuration
Note: The configuration name will be used as a directory name in the file system. Please ensure that it is valid for your platform.
Name: Debug for ARM
Description:
Copy settings from
Existing configuration Debug
Default configuration



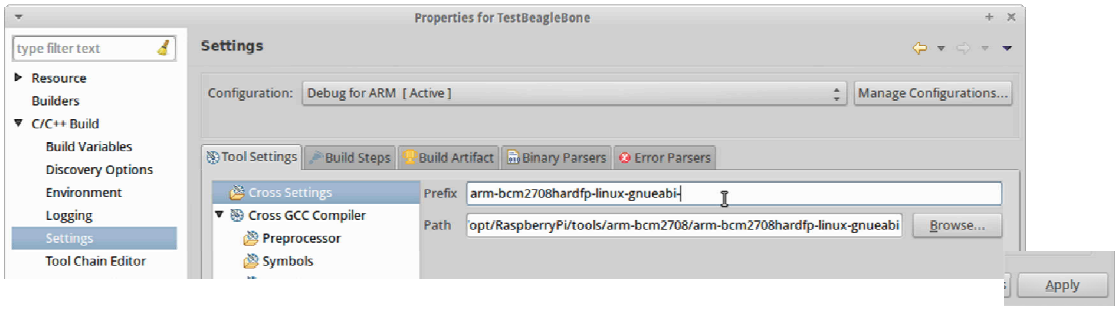
- En Project -> Properties -> C/C++ Build -> Tool Chain Editor: cambiar el toolchain de compilación del proyecto a Cross GCC, y presionar Apply (comprobar que la configuración seleccionada es "Debug for ARM").



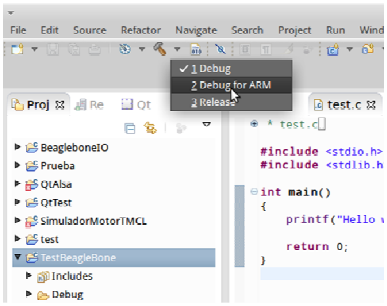
- Sin salirse del diálogo, en C/C++ Build -> Settings -> Tool Settings: cambiar el prefijo y path de compilación, y presionar Apply.

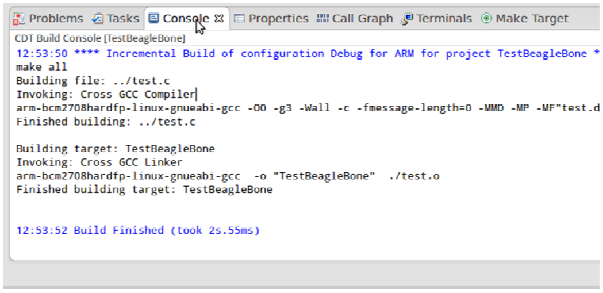
  Prefix: `arm-bcm2708hardfp-linux-gnueabi-`

  Path: `/opt/RaspberryPi/tools/arm-bcm2708/arm-bcm2708hardfp-linux-gnueabi`



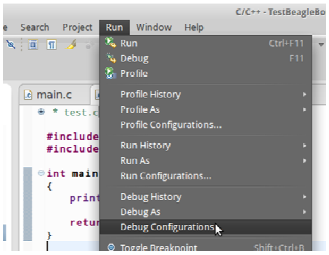
- Cerrar el cuadro de diálogo de propiedades (Accept)
- Recompilar aplicación, asegurándose que se usa la nueva configuración, y revisar consola para comprobar que se utiliza el compilador arm-bcm…-gcc

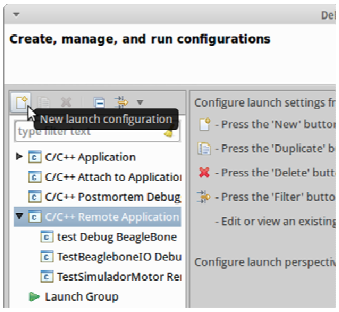



Problems Tasks Console Properties Call Graph Terminals Make Target
CDT Build Console [TestBeagleBone]
12:53:50 **** Incremental Build of configuration Debug for ARM for project TestBeagleBone ****
make all
Building file: ../test.c
Invoking: Cross GCC Compiler
arm-bcm2708hardfp-linux-gnueabi-gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"test.d
Finished building: ../test.c
Building target: TestBeagleBone
Invoking: Cross GCC Linker
arm-bcm2708hardfp-linux-gnueabi-gcc -o "TestBeagleBone" ./test.o
Finished building target: TestBeagleBone
12:53:52 Build Finished (took 2s.55ms)

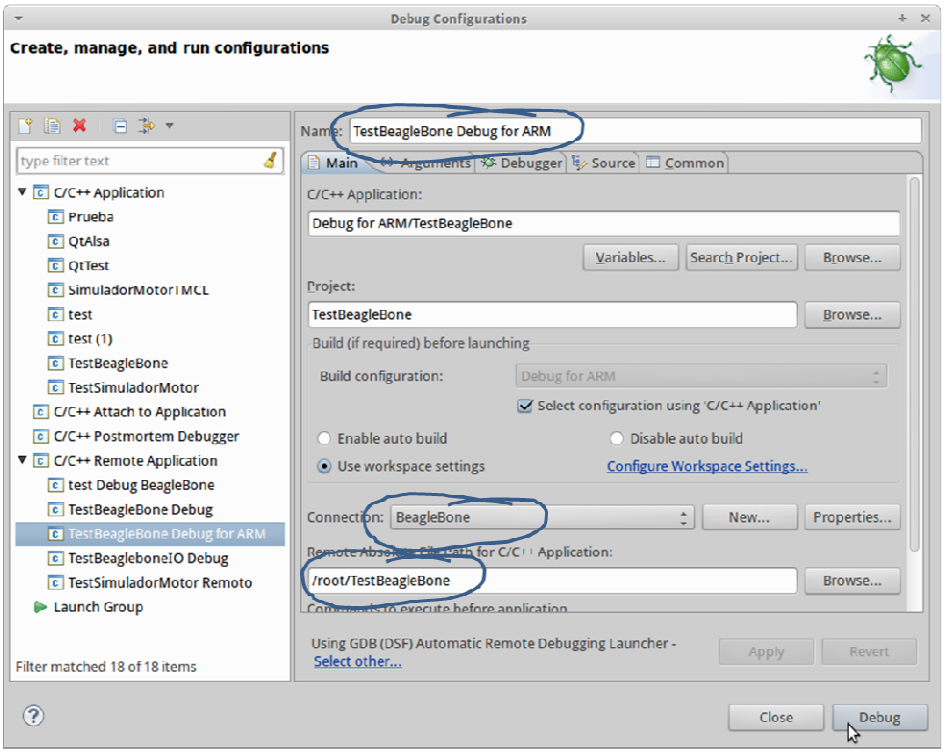
2.2.3. Paso 3: configuración de ejecución remota en BeagleBone

- Crear nueva configuración de depuración en Run -> Debug configurations. Seleccionar C/C++ Remote Application y pulsar botón New.

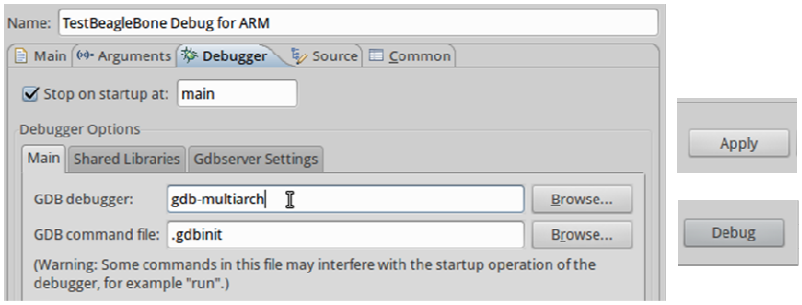




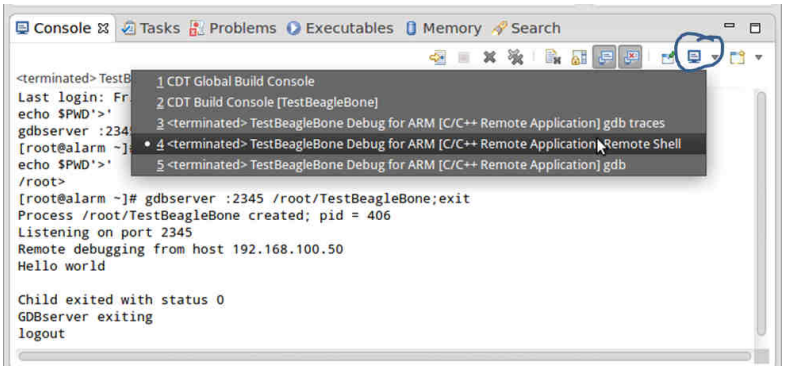
  - En la patilla Main, dar nombre a la nueva configuración de ejecución, elegir la conexión creada anteriormente (BeagleBone) y escribir el lugar donde debe guardarse el ejecutable (/root/nombre de aplicación).



  - En la patilla Debugger, indicar el depurador gdb-multiarch. Eliminar el checkbox Stop on startup si no se desea que al comenzar la depuración se pare en main(). Presionar Apply y Debug para probar ejecución.



- En la ventana Console debe apreciarse la entrada en el Shell del Beaglebone, la conexión del depurador con gdbserver, y la ejecución de la aplicación (seleccionar consola Remote Shell).

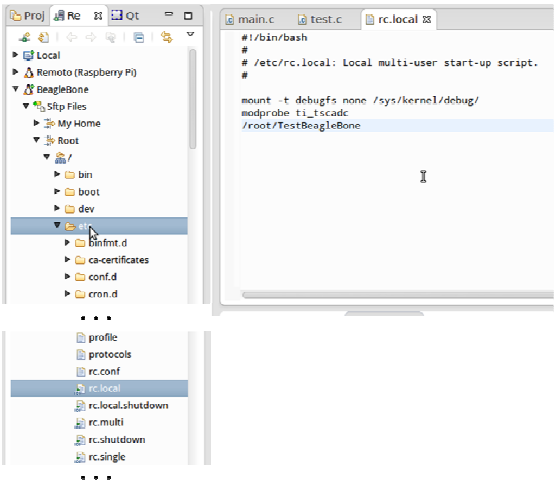


- Cuando se recompile la aplicación, puede de nuevo ejecutarse en modo depuración (paso a paso, visualizando variables) o directamente con Run seleccionando la configuración creada.

2.2.4. Paso 4: instalación definitiva y arranque automático

La aplicación desarrollada con los pasos anteriores se encuentra en /root/TestBeaglebone, y ya puede ser ejecutada directamente sin necesidad del entorno de compilación.

Para conseguir una ejecución automática, entrar en el directorio /etc del sistema remoto (desde la ventana sistema remoto de eclipse, Sftp files, acceder a Root -> / -> etc), y editar el archivo rc.local (hacer doble click para abrir), añadiendo al final la línea que ejecuta el programa. Recordar guardar el archivo modificado.



La próxima vez que se reinicie el BeagleBone se ejecutará el programa directamente.

3. Acceso a dispositivos de E/S

Los dispositivos de E/S habituales de los sistemas Linux (consola, puerto serie, conexión Ethernet, etc.) se acceden en BeagleBone de igual manera que en otros equipos con el mismo S.O.

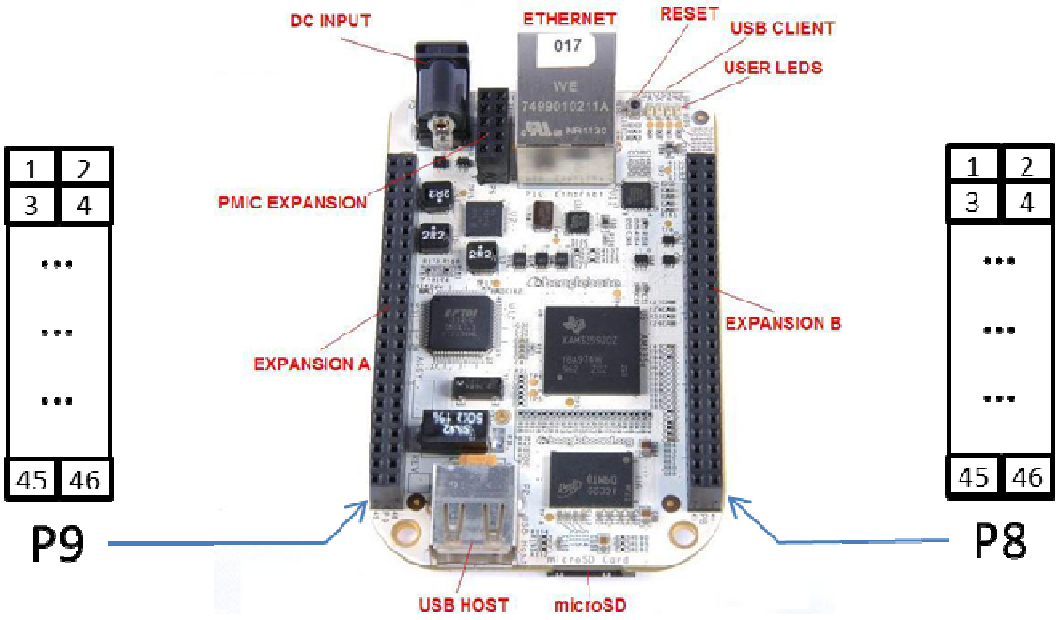
Sin embargo, los dispositivos de E/S más específicos, agrupados bajo el nombre genérico de GPIO – General Purpose Input Output - (E/S digital, entrada analógica, salida PWM) requieren un trabajo más laborioso, que conlleva la lectura de documentación dispersa y en muchos casos sólo existente a partir de respuestas en foros.

A continuación se resumen los requerimientos y procedimientos para esta E/S, y se acompaña una librería de funciones que facilitan su manejo (ver sección 3.3).

3.1. Interfaz hardware

3.1.1. Conectores de expansión P8 y P9

Existen 2 conjuntos de pines de expansión en los conectores P8 y P9, documentados en el manual de referencia (ver 3.1.4). En cada uno de los pines de estos conectores se exponen múltiples señales de la CPU ARM, por lo que existe un multiplexor que permite seleccionar cuál de las señales es accesible en cada momento.



Las señales disponibles en cada pin de estos conectores dependen del modo seleccionado en el multiplexor, según la tabla siguiente:

CONECTOR P8

PIN	MODE							
	0	1	2	3	4	5	6	7
P8_1	GND							
P8_2	GND							
P8_3	gpmc_ad6	mmc1_dat6						GPIO1_6
P8_4	gpmc_ad7	mmc1_dat7						GPIO1_7
P8_5	gpmc_ad2	mmc1_dat2						GPIO1_2
P8_6	gpmc_ad3	mmc1_dat3						GPIO1_3
P8_7	gmpc_advn_ale		timer4					GPIO2_2
P8_8	gpmc_oen_ren		timer7					GPIO2_3
P8_9	gpmc_be0n_cle		timer5					GPIO2_5
P8_10	gpmc_wen		timer6					GPIO2_4
P8_11	gpmc_ad13	lcd_data18	mmc1_dat5	mmc2_dat1	eQEP2B_in			GPIO1_13
P8_12	gpmc_ad12	lcd_data19	mmc1_dat4	mmc2_dat0	EQEP2A_IN			GPIO1_12
P8_13	gpmc_ad9	lcd_data22	mmc1_dat1	mmc2_dat5	ehrpwm2B			GPIO0_23
P8_14	gpmc_ad10	lcd_data21	mmc1_dat2	mmc2_dat6	ehrpwm2_tripzone_in			GPIO0_26
P8_15	gpmc_ad15	lcd_data16	mmc1_dat7	mmc2_dat3	eQEP2_strobe			GPIO1_15
P8_16	gpmc_ad14	lcd_data17	mmc1_dat6	mmc2_dat2	eQEP2_index			GPIO1_14
P8_17	gpmc_ad11	lcd_data20	mmc1_dat3	mmc2_dat7	ehrpwm0_synco			GPIO0_27
P8_18	gpmk_clk_mux0	lcd_memory_clk	gpmc_wait1	mmc2_clk			mcasp0_fsr	GPIO2_1
P8_19	gpmc_ad18	lcd_data23	mmc1_dat0	mmc2_dat4	ehrpwm2A			GPIO0_22
P8_20	gpmc_csn2	gpmc_be1n	mmc1_cmd					GPIO1_31
P8_21	gpmc_csn1	gpmc_clk	mmc1_clk					GPIO1_30
P8_22	gpmc_ad5	mmc1_dat3						GPIO1_5
P8_23	gpmc_ad4	mmc1_dat4						GPIO1_4
P8_24	gpmc_ad1	mmc1_dat1						GPIO1_1
P8_25	gpmc_ad0	mmc1_dat0						GPIO1_0
P8_26	gpmc_csn0							GPIO1_29
P8_27	lcd_vsync	gpmc_a8						GPIO2_22
P8_28	lcd_pclk	gpmc_a10						GPIO2_24
P8_29	lcd_hsync	gpmc_a9						GPIO2_23
P8_30	lcd_ac_bias_en	gpmc_a11						GPIO2_25
P8_31	lcd_data14	gpmc_a18	eQEP1_index	mcasp0_axr1	uart5_rxd		uart5_ctsn	GPIO0_10
P8_32	lcd_data15	gpmc_a19	eQEP1_strobe	mcasp0_ahclkx	mcasp0_axr3		uart5_rtsn	GPIO0_11
P8_33	lcd_data13	gpmc_a17	eQEP1B_in	mcasp0_fsr	mcasp0_axr3		uart4_rtsn	GPIO0_9
P8_34	lcd_data11	gpmc_a15	ehrpwm1B	mcasp0_ahclkr	mcasp0_axr2		uart3_rtsn	GPIO2_17
P8_35	lcd_data12	gpmc_a16	eQEP1A_in	mcasp0_aclkr	mcasp0_axr2		uart4_ctsn	GPIO0_8
P8_36	lcd_data10	gpmc_a14	ehrpwm1A	mcasp0_axr0			uart3_ctsn	GPIO2_16
P8_37	lcd_data8	gpmc_a12	ehrpwm1_tripzone_in	mcasp0_aclkx	uart5_txd		uart2_ctsn	GPIO2_14
P8_38	lcd_data9	gpmc_a13	ehrpwm0_synco	mcasp0_fsx	uart5_rxd		uart2_rtsn	GPIO2_15
P8_39	lcd_data6	gpmc_a6		eQEP2_index				GPIO2_12
P8_40	lcd_data7	gpmc_a7		eQEP2_strobe	pr1_edio_data_out7			GPIO2_13
P8_41	lcd_data4	gpmc_a4		eQEP2A_in				GPIO2_10
P8_42	lcd_data5	gpmc_a5		eQEP2B_in				GPIO2_11
P8_43	lcd_data2	gpmc_a2		ehrpwm2_tripzone_in				GPIO2_8
P8_44	lcd_data3	gpmc_a3		ehrpwm0_synco				GPIO2_9
P8_45	lcd_data0	gpmc_a0		ehrpwm2A				GPIO2_6
P8_46	lcd_data1	gpmc_a1		ehrpwm2B				GPIO2_7

CONECTOR P9

PIN	MODE							
	0	1	2	3	4	5	6	7
P9_1	GND							
P9_2	GND							
P9_3	VDD_3V3EXP							
P9_4	VDD_3V3EXP							
P9_5	VDD_5V							
P9_6	VDD_5V							
P9_7	SYS_5V							
P9_8	SYS_5V							
P9_9	PWR_BUT							
P9_10	RESET_OUT							
P9_11	gpmc_wait0	mii2_crs	gpmc_csn4	rmii2_crs_dv	mmc1_sdcd		uart4_rxd_mux2	GPIO0_30
P9_12	gpmc_be1n	mii2_col	gpmc_csn6	mmc2_dat3	gpmc_dir		mcasp0_aclkr_mux3	GPIO1_28
P9_13	gpmc_wpn	mii2_rxerr	gpmc_csn5	rmii2_rxerr	mmc2_sdcd		uart4_txd_mux2	GPIO0_31
P9_14	gpmc_a2	mii2_txd3	rgmii2_td3	mmc2_dat1	gpmc1_a18		ehrpwm1A_mx1	GPIO1_18
P9_15	gpmc_a0	gmii2_txen	rmii2_tctl	mii2_txen	gpmc_a16		ehrpwm1_tripzone_input	GPIO1_16
P9_16	gpmc_a3	mii2_txd2	rgmii2_td2	mmc2_dat2	gpmc_a19		ehrpwm1B_mx1	GPIO1_19
P9_17	spi0_cs0	mmc2_sdwp	I2C1_SCL	ehrpwm0_synci				GPIO0_5
P9_18	spi0_d1	mmc1_sdwp	I2C1_SDA	ehrpwm0_tripzone				GPIO0_4
P9_19	uart1_rtsn	timer5	dcan0_rx	I2C2_SCL	spi1_cs1			GPIO0_13
P9_20	uart1_ctsn	timer6	dcan0_tx	I2C2_SDA	spi1_cs0			GPIO0_12
P9_21	spi0_d0	uart2_txd	I2C2_SCL	ehrpwm0B			EMU3_mux1	GPIO0_3
P9_22	spi0_sclk	uart2_rxd	I2C2_SDA	ehrpwm0A			EMU2_mux1	GPIO0_2
P9_23	gpmc_a1	gmii2_rxdv	rgmii2_rxdv	mmc2_dat0	gpmc_a17		ehrpwm0_synco	GPIO1_17
P9_24	uart1_txd	mmc2_sdwp	dcan1_rx	I2C1_SCL				GPIO0_15
P9_25	mcasp0_ahclkx	eQEP0_strobe	mcasp0_axr3	mcasp1_axr1	EMU4_mux2			GPIO3_21
P9_26	uart1_rxd	mmc1_sdwp	dcan1_tx	I2C1_SDA				GPIO0_14
P9_27	mcasp0_fsr	eQEP0B_in	mcasp0_axr3	mcasp1_fsx	EMU2_mux2			GPIO3_19
P9_28	mcasp0_ahclkr	ehrpwm0_synci	mcasp0_axr2	spi1_cs0	eCAP2_in_PWM2_out			GPIO3_17
P9_29	mcasp0_fsx	ehrpwm0B		spi1_d0	mmc1_sdcd_mux1			GPIO3_15
P9_30	mcasp0_axr0	ehrpwm0_tripzon		spi1_d1	mmc2_sdcd_mux1			GPIO3_16
P9_31	mcasp0_aclkx	ehrpwm0A		spi1_sclk	mmc0_sdcd_mux1			GPIO3_14
P9_32	VDD_ADC_1.8V							
P9_33	AIN4							
P9_34	ADC_GND							
P9_35	AIN6							
P9_36	AIN5							
P9_37	AIN2							
P9_38	AIN3							
P9_39	AIN0							
P9_40	AIN1							
P9_41	xdma_event_intr1		tclkin	clkout2	timer7_mux1		EMU3_mux0	GPIO0_20
P9_42	eCAP0_in_PWM0_out	uart3_txd	spi1_cs1	pr1_ecap0_ecap_capin_apwm_o	spi1_sclk	mmc0_sdwp	xdma_event_intr2	GPIO0_7
P9_43	GND							
P9_44	GND							
P9_45	GND							
P9_46	GND							

3.1.2. E/S digitales

Se pueden utilizar múltiples pines para E/S digital a través de los conectores P8 y P9 (todos los indicados con GPIOx_xx en el modo 7. Seleccionar para estos fines aquellos pines que no se piensan utilizar en otros (ej. salida PWM).

¡¡¡ OJO!!! Estos pines funcionan a **3.3V**; se daña la placa si se utilizan con una tensión mayor. Cuando están configuradas como salida, la máxima corriente que se debe solicitar es de **6 mA**.

Cuando se configura un pin para E/S digital (modo 7), se puede seleccionar si se desea entrada o salida, y si se activa una resistencia interna pull-up o pull-down, que permite que tengan un valor predefinido en vacío. Todo ello se configura en los bits 4-7 del modo, según la siguiente tabla:

BIT	7	6	5	4	3	2	1	0
Valor	0	0	1=in 0=out	0=pulldown 1=pullup	0=pullenable 1=pulldisable	1	1	1

La activación y configuración de estos pines se realiza mediante las funciones de la librería suministrada (ver apartado 3.3):

- int InitDigitalIO(enumDIOpin pin,enumDIOtype io,enumDIOpull pull);
- int ReadDigitalInput(int pinId);
- int WriteDigitalOutput(int pinId,int value);
- int EndDigitalIO(int pinId);

3.1.3. Entradas analógicas

Existen 7 entradas analógicas están disponibles en los pines AINxx en el conector P9, con 12 bits de resolución (valores 0 a 4095), y frecuencia de muestreo máxima de 100Ks/s (aunque ésta se puede reducir debido al driver de acceso).

¡¡¡ OJO!!! Estas entradas tienen un rango de tensión de **0V a 1.8V**; se daña la placa si se utilizan con una tensión mayor.

Las entradas analógicas se leen mediante las funciones siguientes de la librería suministrada (ver apartado 3.3):

- int InitAnalogInput(enumAIpin pin);
- int ReadAnalogInput(int pinId);
- int EndAnalogInput(int pinId);

3.1.4. Salidas PWM

Existen 3 salidas PWM, que se activan seleccionando el modo adecuado en los pines correspondientes:

- EHRPWM0A: pin P9_22 (modo 3) ó pin P9_31 (modo 1).
- EHRPWM0B: pin P9_21 (modo 3) ó pin P9_29 (modo 1).
- EHRPWM1A: pin P8_36 (modo 2) ó pin P9_14 (modo 6).
- EHRPWM1B: pin P8_34 (modo 2) ó pin P9_16 (modo 6).
- EHRPWM2A: pin P8_19 (modo 4) ó pin P8_45 (modo 3).
- EHRPWM2B: pin P8_13 (modo 4) ó pin P8_46 (modo 3).

Para que funcionen las salidas PWM, es necesario previamente activar el reloj de las mismas, escribiendo directamente en las posiciones adecuadas del mapa de E/S (ya realizado en librería disponible).

¡¡¡ OJO!!! Estos pines funcionan a **3.3V**; se daña la placa si se utilizan con una tensión mayor. Cuando están configuradas como salida, la máxima corriente que se debe solicitar es de **6 mA**.

Las salidas PWM se utilizan mediante las funciones siguientes de la librería suministrada (ver apartado 3.3):

- int InitPWMOutput(enumPWMpin pin,int freq);
- int WritePWMOutput(int pinId,float duty);
- int EndPWMOutput(int pinId);

3.2. Acceso a la E/S mediante Linux fs (filesystem)

Esta sección ya está implementada en la librería disponible, sólo es a título informativo. Si sólo se desea utilizar dicha librería, pasar directamente a la sección 3.3.

Para acceder a los dispositivos de E/S se utilizan los drivers por defeco suministrados en Linux. Para que éstos funcionen, debe montarse el filesystem con la orden siguiente (ya realizado en /etc/rc.local):

```
mount -t debugfs none /sys/kernel/debug/
```

Cada pin de los conectores de expansión tiene un dispositivo asociado, según la tabla siguiente:

GPIO1_6	"gpmc_ad6"	GPIO1_7	"gpmc_ad7"
GPIO1_2	"gpmc_ad2"	GPIO1_3	"gpmc_ad3"
TIMER4	"gpmc_advn_ale"	TIMER7	"gpmc_oen_ren"
TIMER5	"gpmc_ben0_cle"	TIMER6	"gpmc_wen"
GPIO1_13	"gpmc_ad13"	GPIO1_12	"gpmc_ad12"
EHRPWM2B	"gpmc_ad9"	GPIO0_26	"gpmc_ad10"
GPIO1_15	"gpmc_ad15"	GPIO1_14	"gpmc_ad14"
GPIO2_1	"gpmc_clk"	EHRPWM2A	"gpmc_ad8"
GPIO1_31	"gpmc_csn2"	GPIO1_30	"gpmc_csn1"
GPIO1_5	"gpmc_ad5"	GPIO1_4	"gpmc_ad4"
GPIO1_1	"gpmc_ad1"	GPIO1_0	"gpmc_ad0"
GPIO1_29	"gpmc_csn0"	GPIO2_22	"lcd_vsync"
GPIO2_24	"lcd_pclk"	GPIO2_23	"lcd_hsync"
GPIO2_25	"lcd_ac_bias_en"	UART5_CTSN	"lcd_data14"
UART5_RTSN	"lcd_data15"	UART4_RTSN	"lcd_data13"
UART3_RTSN	"lcd_data11"	UART4_CTSN	"lcd_data12"
UART3_CTSN	"lcd_data10"	UART5_TXD	"lcd_data8"
UART5_RXD	"lcd_data9"	GPIO2_12	"lcd_data6"
GPIO2_13	"lcd_data7"	GPIO2_10	"lcd_data4"
GPIO2_11	"lcd_data5"	GPIO2_8	"lcd_data2"
GPIO2_9	"lcd_data3"	GPIO2_6	"lcd_data0"
GPIO2_7	"lcd_data1"	UART4_RXD	"gpmc_wait0"
GPIO1_28	"gpmc_ben1"	UART4_TXD	"gpmc_wpn"

EHRPWM1A	"gpmc_a2"	GPIO1_16	"gpmc_a0"
EHRPWM1B	"gpmc_a3"	I2C1_SCL	"spi0_cs0"
I2C1_SDA	"spi0_d1"	I2C2_SCL	"uart1_rtsn"
I2C2_SDA	"uart1_ctsn"	UART2_TXD	"spi0_d0"
UART2_RXD	"spi0_sclk"	GPIO1_17	"gpmc_a1"
UART1_TXD	"uart1_txd"	GPIO3_21	"mcasp0_ahclkx"
UART1_RXD	"uart1_rxd"	GPIO3_19	"mcasp0_fsr"
SPI1_CS0	"mcasp0_ahclkr"	SPI1_D0	"mcasp0_fsx"
SPI1_D1	"mcasp0_axr0"	SPI1_SCLK	"mcasp0_aclkx"
CLKOUT2	"xdma_event_intr1"	GPIO0_7	"ecap0_in_pwm0_out"
LED_USR0	"gpmc_a5"	LED_USR1	"gpmc_a6"
LED_USR2	"gpmc_a7"	LED_USR3	"gpmc_a8"

¡¡¡ OJO!!! Todos los archivos del filesystem para E/S son accesibles en modo texto. Todos los textos a escribir deben estar terminados en retorno de carro '\n'.

3.2.1. Manejo de E/S digital en pines P8 y P9 y LEDs de placa

Para configurar un pin para E/S digital, se deben realizar los siguientes pasos, leyendo o escribiendo en los archivos indicados del filesystem

- Establecer el modo correspondiente al pin (bits indicados en apartado 3.1.2) escribiendo su valor en hexadecimal en el archivo "/sys/kernel/debug/omap_mux/%s", donde %s debe sustituirse por el nombre asociado al pin según la tabla del apartado 3.2.
- Obtener el nº de pin, por medio del cálculo siguiente a partir de la GPIO asociada (columna de modo 7):

  GPIOx_yy ⇨ pin=(x*32)+yy
- Exportar el pin al filesystem, escribiendo su nº (en decimal) en el archivo "/sys/class/gpio/export".
- Establecer la dirección (E ó S) escribiendo "in" ó "out" en el archivo "/sys/class/gpio/gpio%d/direction", donde %d debe sustituirse por el nº de pin.
- Si se trata de una salida, escribir "1" ó "0" en el archivo "/sys/class/gpio/gpio%d/value", donde %d debe sustituirse por el nº de pin.
- Si se trata de una salida de LED de la placa (USR0 a USR3), el archivo a escribir para establecer su valor debe ser "/sys/devices/platform/leds-gpio/leds/beaglebone::usr%d/brightness", en lugar del anterior, donde %d debe sustituirse por el nº de led (0 a 3).
- Si se trata de una entrada, leer 2 caracteres del archivo "/sys/class/gpio/gpio%d/value", donde %d debe sustituirse por el nº de pin. El 1er carácter leído será '1' ó '0' en función del valor de la entrada.

3.2.2. Manejo de entradas analógicas

Las 7 entradas analógicas (AIN0 a AIN6) no están multiplexadas, por lo que simplemente hay que leer del archivo correspondiente del filesystem. Para que funcionen, es necesario cargar el siguiente módulo de drivers (ya realizado en /etc/rc.local):

```
modprobe ti_tscadc
```

- Lectura de entrada analógica: leer texto del archivo "/sys/devices/platform/omap/tsc/ain%1d", donde %d se sustituye por el nº de entrada, y convertir a valor entero (valores de 0 a 4095).

3.2.3. Manejo de salidas PWM

Las 3 salidas analógicas en modo PWM están multiplexadas con pines de GPIO, por lo que debe tenerse cuidado de seleccionarlas adecuadamente. Para utilizarlas, en primer lugar hay que activar los relojes correspondientes, lo que se realiza escribiendo directamente en el mapa de E/S con la función mmap() en los offset adecuados (ver archivo "o_pwm.c" en el código fuente de la librería).

Las patillas A y B de cada una de las salidas PWM se acceden en archivos del directorio "/sys/class/pwm/ehrpwm.%d:%d/%s", donde el 1er %d se sustituye por el nº de la PWM; el 2º %d se sustituye por 0 para la salida A, o por 1 para la salida B; y el %s se sustituye por el nombre de archivo del filesystem según la tabla del apartado 3.2.

A continuación, se deben realizar los siguientes pasos:

- Escribir "1" en el archivo "request" de dicho directorio.
- Escribir "1" en el archivo "run" de dicho directorio.
- Escribir el texto correspondiente al valor de frecuencia deseada para el PWM (en Hz) en el archivo "period_freq" de dicho directorio.
- Cambiar el porcentaje de duty deseado escribiendo como texto un valor entero de 0 a duty*periodo (en ns) en el archivo "duty_ns" de dicho directorio.

3.3. Librería de E/S

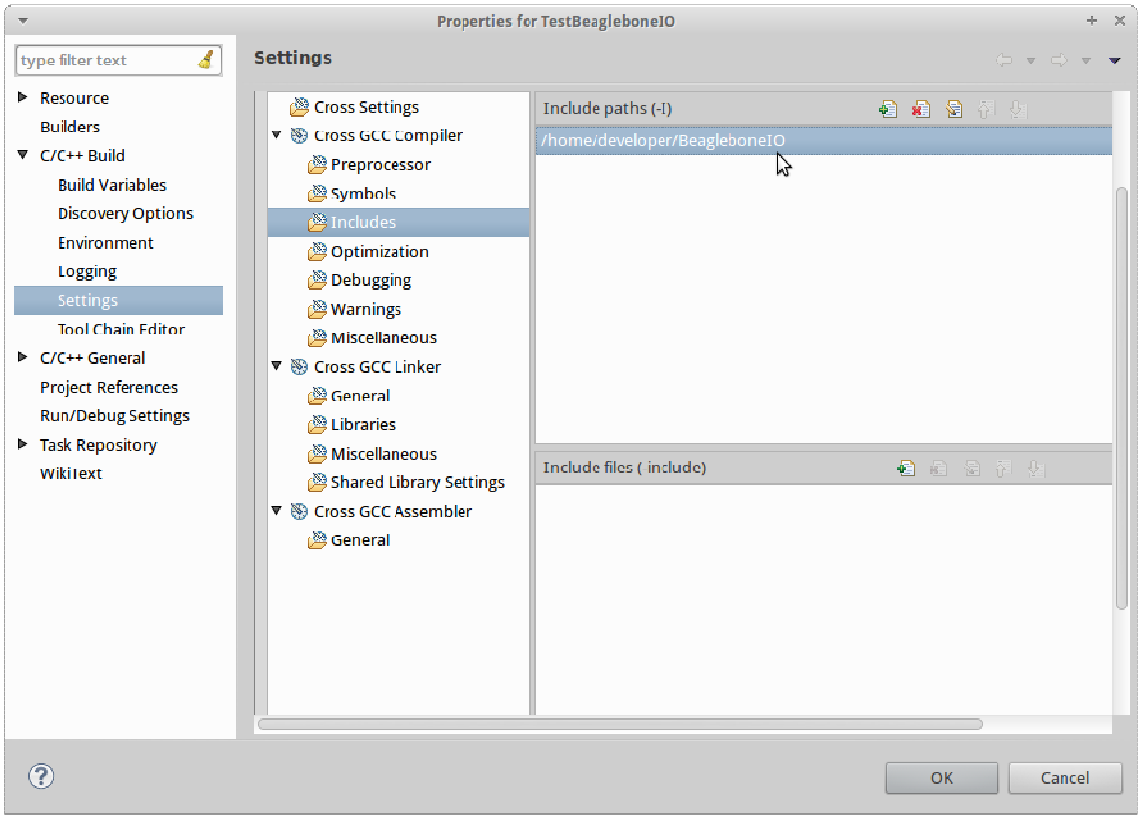
Para facilitar el trabajo con los pines de E/S, se ha desarrollado una librería que permite realizar las distintas operaciones de forma sencilla.

Para utilizarla, deben descargarse de la página web los archivos "io_defs.h" y "libBeagleboneIO.a", y guardarlos en un directorio del sistema Linux (ej. "/home/developer/BeagleBoneIO".

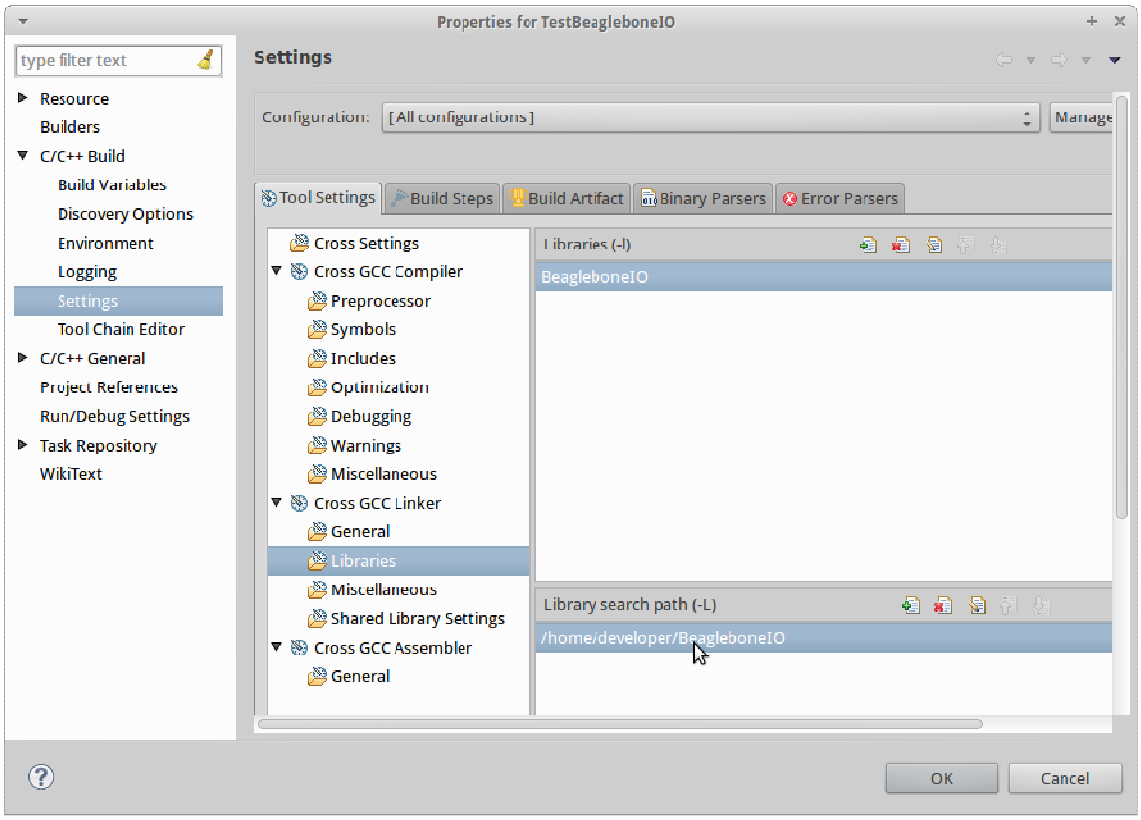
3.3.1. Configuración de proyecto

En el proyecto eclipse, debe configurarse el uso de la librería con los siguientes pasos:

- Establecer directorio de inclusión adicional: entrar en Project -> Properties -> C/C++ Build -> Settings -> Tool Settings -> Cross GCC Compiler -> Includes -> Include Path, y añadir el directorio de la librería (seleccionar previamente todas las configuraciones).



- Sin salir del diálogo, establecer directorio de librerías: Tool Settings -> Cross GCC Linker -> Libraries -> Library Search Path, y añadir la librería BeagleBoneIO en Librairies.



3.3.1. Manejo de E/S digital

Las funciones para manejo de E/S digital son las siguientes:

int InitDigitalIO(enumDIOpin pin,enumDIOtype io,enumDIOpull pull);

Descripción:

Configura un pin de las patillas de expansión como salida digital

Parámetros:

pin = Nº de pin del conector de expansión a configurar. Indicar con PIN_Px_yy, donde x es 8 ó 9, e yy el nº de pin (2 dígitos). También se pueden utilizar los valores LED_USRz, con z de 0 a 3, sólo para salida.

io = Modo de funcionamiento: DIO_INPUT ó DIO_OUTPUT.

pull = Resistencia interna: DIO_PULLUP, DIO_PULLDOWN ó DIO_NOPULL.

Valor devuelto:

Entero con identificador a utilizar en las siguientes llamadas, o -1 si hay error.

Ejemplo:

Configurar el pin 5 del conector 8 para entrada con resistencia pull up:

```
int p85;
p85=InitDigitalIO(PIN_P8_05,DIO_INPUT,DIO_PULLUP);
```

int ReadDigitalInput(int pinId);

Descripción:

Lee el valor actual de un pin de E/S configurado como entrada digital.

Parámetros:

pinId = Identificador de pin devuelto por la función InitDigitalIO().

Valor devuelto:

Valor del bit de entrada (0 ó 1), o negativo si no se ha podido leer.

Ejemplo:

Leer el valor del pin P8_5 configurado anteriormente:

```
valor=ReadDigitalInput(p85);   // Donde valor debe ser un int
```

int WriteDigitalInput(int pinId,int value);

Descripción:

Escribe el valor deseado en un pin de E/S configurado como salida digital.

Parámetros:

pinId = Identificador de pin devuelto por la función InitDigitalIO().

value =Valor a establecer (0 ó 1).

Valor devuelto:

Valor escrito (0 ó 1), o valor negativo si no se ha podido escribir.

Ejemplo:

Configurar el led USR1 como salida digital y escribir un 1:

```
int led1;
led1=InitDigitalIO(LED_USR1,DIO_OUTPUT,DIO_NOPULL);
WriteDigitalOutput(led1,1);
```

int EndDigitalIO(int pinId);

Descripción:

Desconfigura el pin de E/S deseado. El identificador ya no es utilizable en siguientes llamadas.

Parámetros:

pinId = Identificador de pin devuelto por la función InitDigitalIO().

Valor devuelto:

0 si correcto, negativo si incorrecto.

Ejemplo:

```
int p85;
p85=InitDigitalIO(PIN_P8_05,DIO_INPUT,DIO_PULLUP)
... usar p85 ...
EndDigitalIO(p85);
... no usar más p85 ...
```

3.3.2. Manejo de entrada analógica

Las funciones para manejo de entradas analógicas son las siguientes:

int InitAnalogInput(enumAIpin pin);

Descripción:

Habilita entrada analógica en pin del conector de expansión P9.

Parámetros:

pin = Nº de pin del conector de expansión a configurar. Indicar con PIN_P9_yy, donde yy es el nº de pin (2 dígitos). También se pueden utilizar los valores AIN0 a AIN6.

Valor devuelto:

Entero con identificador a utilizar en las siguientes llamadas, o -1 si no es posible la configuración.

Ejemplo:

Configurar el pin 37 del conector 9 (AIN2) para entrada analógica:

```
int p9_37;
p9_37=InitAnalogInput(PIN_P9_37);
```

int ReadAnalogInput(int pinId);

Descripción:

Lee el valor actual de un pin de entrada analógica.

Parámetros:

pinId = Identificador de pin devuelto por la función InitAnalogInput ().

Valor devuelto:

Valor 0 a 4095, o negativo si no se ha podido leer.

Ejemplo:

Leer el valor del pin P9_37 configurado anteriormente:

```
valor=ReadAnalogInput(p9_37); // Donde valor debe ser un int
```

int EndAnalogInput(int pinId);

Descripción:

Desconfigura el pin de entrada analógica. El identificador ya no es utilizable en siguientes llamadas.

Parámetros:

pinId = Identificador de pin devuelto por la función InitAnalogInput ().

Valor devuelto:

0 si correcto, negativo si incorrecto.

Ejemplo:

```
int p9_37;
p9_37=InitAnalogInput(PIN_P9_37);
… usar p9_37…
EndAnalogInput (p9_37);
… no usar más p9_37…
```

3.3.3. Manejo de salida PWM

Las funciones para manejo de salidas PWM son las siguientes:

int InitPWMOutput(enumPWMpin pin,int freq);

Descripción:

Habilita salida PWM en la patilla del conector de expansión P8 ó P9.

Parámetros:

pin = Nº de pin a configurar. Indicar con PIN_Px_yy, donde x es el nº de conector (8 ó 9) e yy es el nº de pin (2 dígitos). Ver pines válidos en 3.1.4.

Valor devuelto:

Entero con identificador a utilizar en las siguientes llamadas, o -1 si hay error.

Ejemplo:

Configurar el pin 13 del conector 8 (EHRPWM2B) para salida PWM con frecuencia 500 Hz:

```
int pwm;
pwm= InitPWMOutput(PIN_P8_13,500);
```

int WritePWMOutput(int pinId,float duty);

Descripción:

Escribe el valor de duty en la salida PWM.

Parámetros:

pinId = Identificador de pin devuelto por la función InitPWMOutput().

duty = Valor de duty deseado (0-1)

Valor devuelto:

0 si correcto, o negativo si no se ha podido escribir.

Ejemplo:

Escribir un duty del 25% en el pin P8_13 configurado anteriormente:

```
WritePWMOutput (pwm,0.25F);
```

int EndPWMOutput (int pinId);

Descripción:

Desconfigura el pin de salida PWM. El identificador ya no es utilizable en siguientes llamadas.

Parámetros:

pinId = Identificador de pin devuelto por la función InitPWMOutput().

Valor devuelto:

0 si correcto, negativo si incorrecto.

Ejemplo:

```
int pwm;
pwm= InitPWMOutput(PIN_P8_13,500);
… usar pwm…
EndPWMOutput (pwm);
… no usar más pwm …
```

3.4. Fuentes de documentación

- Hardware: BeagleBone System Reference Manual: http://beagleboard.org/static/beaglebone/latest/Docs/Hardware/BONE_SRM.pdf
- Uso de los pines de GPIO: http://www.nathandumont.com/node/250
- Activación de leds integrados en placa: http://www.drdobbs.com/embedded-systems/beaglebone-leds/240001423
- E/S digital y PWM: http://www.gigamegablog.com/2012/03/16/beaglebone-coding-101-buttons-and-pwm/